Claims

1. A method of applying two effects to an image, the method comprising the steps of
   - using a first microprocessor to apply a first effect to a first frame of said image,
   - using a second microprocessor to apply a second effect to said first-effected frame, applying said first effect to a next frame by said first microprocessor approximately during the time that said second microprocessor is applying said second effect to said first-effected frame.
2. The method of claim 1 wherein the first microprocessor is a CPU and the second microprocessor is a GPU.
3. The method of claim 1 wherein the first microprocessor is a GPU and the second microprocessor is a CPU.
4. The method of claim 1 wherein said first microprocessor applies said first effect by emulating said a GPU.
5. The method of claim 1 where said second effect is a color conversion.
6. The method of claim 1 wherein a said first microprocessor is a CPU if said first effect is better suited to said CPU than an available GPU.
7. The method of claim 1 wherein said first microprocessor is a GPU if said first effect is better suited to said GPU than an available CPU.
8. A method of creating an image, said image represented by an image graph, said image graph comprising one or more GPU programs, inputs to those programs and outputs from those programs, the method comprising the steps of:
   - optimizing said image graph by running software on a CPU;
   - compiling said image graph by running software on said CPU;
   - rendering said image graph by running said compiled image graph on a GPU, yielding a rendered image.
9. The method of claim 8, wherein the step of optimizing includes the step of using a cache look-up to see if said rendered image is already in cache.
10. The method of claim 8 wherein the step of optimizing includes the step of using a cache look-up to see if said image graph has already been optimized and is in a memory.

11. The method of claim 8, wherein the step of optimizing includes the step of Calculating an intersection, said intersection representing an area where said rendered image is both defined by said image graph and part of a region requested by a process running on said CPU that has requested creation of said image.
12. The method of claim 9, wherein the step of optimizing includes the step of Calculating an intersection, said intersection representing an area where said rendered image is both defined by said image graph and part of a region requested by a process running on said CPU that has requested creation of said image.
13. The method of claim 10, wherein the step of optimizing includes the step of Calculating an intersection, said intersection representing an area where said rendered image is both defined by said image graph and part of a region requested by a process running on said CPU that has requested creation of said image.
14. The method of claim 11 further comprising the step of, using said calculated intersection to limit the number of pixels that require calculation during said rendering on said GPU.
15. The method of claim 12 further comprising the step of, using said calculated intersection to limit the number of pixels that require calculation during said rendering on said GPU
16. The method of claim 13 further comprising the step of, using said calculated intersection to limit the number of pixels that require calculation during said rendering on said GPU
17. The method of claim 11 further comprising the step of, using said calculated intersection to limit the amount of memory necessary for storing said rendered image.
18. The method of claim 12 further comprising the step of, using said calculated intersection to limit the amount of memory necessary for storing said rendered image.
19. The method of claim 13 further comprising the step of, using said calculated intersection to limit the amount of memory necessary for storing said rendered image.
20. The method of claim 8 wherein said step of optimizing comprises the additional steps of
    - using a cache to determine if said rendered image is available in memory.
    - Using the CPU to perform ROI /DOD intersections with respect to one or more of said GPU programs;
    - Using the CPU to determine if GPU programs may be combined;
    - Using the CPU to determine if said GPU is capable of running a program that has been created by combining two other GPU programs

21. The method of claim 8 wherein said step of optimizing comprises the additional steps of
    - Using the CPU to perform ROI/DOD intersections with respect to one or more of said GPU programs;
    - Using the CPU to determine if GPU programs may be combined;
    - Using the CPU to determine if said GPU is capable of running a program that has been created by combining two other programs
22. The method of claim 8 wherein said step of optimizing comprises the additional steps of
    - Using the CPU to determine if GPU programs may be combined;
    - Using the CPU to determine if said GPU is capable of running a program that has been created by combining two other programs
23. A method for creating a rendered polygon, the method comprising the steps of::
    - Under control of a CPU, receiving a request to render a polygon;
    - Under control of said CPU, creating a representation of said rendered polygon comprising a root GPU program and its relationship with other GPU programs, their inputs and outputs;
    - Under control of said CPU, starting with the root GPU program, calling the following groups of objects for each GPU program that must be run in order that the root GPU program may run to render said polygon;
        - one or more objects for analyzing whether two GPU programs may be combined, and performing a combination if said analysis is positive;
        - one or more objects for performing DOD/ROI optimization;
        - one or more objects for creating a buffer and causing a GPU to render an image to that buffer by running a GPU program
24. The method of claim 23 wherein said representation of said rendered polygon is a graph.
25. The method of claim 23 wherein said representation of said rendered polygon is a low-level graph.
26. The method of claim 23 wherein said representation of said rendered polygon is a high-level graph.
27. The method of claim 23 further wherein an application program under CPU control makes said request to render said polygon.

28. The method of claim 23 wherein said one or more objects for analyzing whether two GPU programs may be combined, and performing a combination if said analysis is positive, are recursively called until no more combinations are possible.
29. The method of claim 23 wherein DOD/ROI optimization comprises intersecting the ROI with the output DOD for a GPU program.
30. The method of claim 23 wherein a cache is used in order to find the result of running a GPU program, without running that GPU program.
31. A method for creating a rendered image, the method comprising the steps of:
    - Under control of said CPU, creating a graph representing said rendered image;
    - Under control of said CPU, starting with a root node in said graph, calling the following groups of objects for each node that must be calculated in order that the root node may be calculated;
        - one or more objects for analyzing whether two GPU programs may be combined, and performing a combination if said analysis is positive;
        - one or more DOD/ROI objects for performing DOD/ROI optimization;
        - one or more objects for creating a buffer and causing a GPU to render an image to that buffer by running a GPU program.
32. The method of claim 31 wherein said graph is a low-level graph.
33. The method of claim 31 wherein said graph is a high-level graph.
34. The method of claim 31 further comprising the step of receiving a request to render said image.
35. The method of claim 34 further comprising the step of an application program under CPU control making said request to render said image.
36. The method of claim 31 wherein said one or more objects for analyzing whether two GPU programs may be combined, and performing a combination if said analysis is positive, are recursively called until no more combinations are possible
37. The method of claim 31 wherein DOD/ROI optimization comprises intersecting the ROI with the output DOD for a GPU program.
38. The method of claim 31 wherein a cache is used in order to find the result of computed node without computing that node.

39. A computer-readable medium having computer executable instructions for performing the method recited in any one of claims 1, 8, 20, 23 or 31.